Integrating Publish-Subscribe and Peer-to-Peer Communication for Modeling & Simulation of Moving Entities

Fernando J. Barros
University of Coimbra
Department of Informatics Engineering
3030 Coimbra
PORTUGAL
E-mail: barros@dei.uc.pt

Abstract

The representation of spatially moving entities is commonly achieved using publish/subscribe communication (PSC) supported by some frameworks, like the High Level Architecture (HLA). PSC, however, can become complex and inefficient since it requires the definition of regions of interest and/or it generates false positive messages that need to be filtered and discarded, impacting negatively simulation performance. On the other hand, conventional static peer-to-peer communication (P2PC), though usually more efficient, does not have the required flexibility to represent mobile entities. In this paper we present the integration of PSC and P2PC styles under the hierarchical and modular dynamic topology paradigm supported by the Hybrid Flow System Specification (HFSS) formalism. This unification is achieved using runtime topology adaptation involving the dynamic creation/deletion of links in order to capture the current interactions between entities. The proposed architecture combines the advantages of PSC and P2PC, enabling a flexible simulation framework. Two types of PSC are described: the traditional push style, based on discrete events, and a novel pull style based on sampling. The benefits of this approach are demonstrated through the modeling of an air-defense scenario that is described in the JUSE-*HFSS modeling and simulation framework, an implementation of the HFSS formalism.*

1 Introduction

The representation of spatially moving entities is commonly achieved using publish/subscribe communication (PSC) being supported by some frameworks, like the High Level Architecture (HLA) [9]. PSC is based on the *observer design pattern* [8], and it offers several advantages for the simulation of moving entities, including support for model reuse and the ability to handle the interaction of an arbitrary number of entities. Taking as an example an air-defense system, PSC provides a solution for enabling the communication of a radar with an arbitrary number of aircrafts. An aircraft entering simulation becomes automatically linked to all radars without any modification in the existing radars and aircrafts. Thus, model reuse is preserved while keeping a great flexibility to support moving entities. PSC communication has, however, several drawbacks. Information can only be pushed, making information pulling, like getting the position of an aircraft, more difficult to achieve since it requires two pull-messages to be accomplished. Another limitation is the low efficiency of PSC. When entities are far apart they cannot actually communicate, requiring an intensive filtering mechanism [11] in order to check and discard the messages that cannot go through since entities are out of sensor ranges. Message filtering can be achieved using HLA Data Distribution Management (DDM) and spatial partitioning, but it involves modeling complexity and performance penalties. In fact, since PSC is essentially a broadcast protocol, it can become unnecessarily complex for establishing the communication between two entities that are known to be connected. In this case, we should be able to represent the required interactions using peer-to-peer communication (P2PC), avoiding filtering and spatial distribution management. Examples include the interaction between an aircraft and the beam of a rotating radar when both entities are already within range. Another example is the interaction of a missile and a target. Given that missiles are launched after a target has been selected, one should be able to exploit the peer-to-peer interaction that is easier to represent and more efficient than the alternative solution based on PSC. In particular, P2PC avoids the regular updating of regions of interest (ROIs) required by spatial partitioning for representing moving entities, and the search involved in finding the overlapping of these regions.

The HLA adheres to the discrete event paradigm, offering little support for describing continuous signals as required for mobile entities. Aircrafts, for example, can be described by their trajectories, and PSC would require aircrafts to push their position to radars. However, since radars sample targets at their own rates, PSC can hardly be used to achieve a faithful model of radars. Since the position of an entity is a continuous magnitude often known in advance or described, for example, by ordinary deferential equations (ODEs), it becomes necessary to use a function generator or a numerical integrator to compute entity position. Most of models involving mobile entities are actually hybrid, exhibiting both continuous and discrete behaviors. In an air-defense scenario both radar and aircrafts can be described by continuous trajectories. However, radar echoes are discrete events signals associated with aircraft detection. The combination of both types of signals can be achieved with zero-cross detectors that signal an event when two continuous signals match according to some criteria. We consider that the modeling of complex hybrid systems requires the ability to represent trajectory generators (for known paths), numerical integrators and threshold crossers (for event detection). These requirements are difficult to be fulfilled by the discrete event representation underlaying the HLA.

In this paper we exploit the Hybrid Flow System Specification (HFSS), a modeling formalism aimed to represent hybrid systems with a dynamic typology, to achieve a unified representation of publish-subscribe and peer-to-peer communication styles. This formalism provides the integration of discrete event systems with sampling-based systems enabling a unified representation of hybrid models. To show formalism expressiveness, we model a simplified air-defense scenario in JUSE-HFSS M&S framework, an implementation of HFSS, and we present the advantages of our approach when compared with the more conventional HLA representation.

The paper is organized as follows. Section 2 presents the JUSE-HFSS modeling & simulation environment. Section 3 provides an overview of the HLA. The integration of publish-subscribe and peer-to-peer communication styles is discussed in Section 4. Section 5 describes a simplified air-defense scenario modeled in the HFSS formalism. Simulation results are presented in Section 6. Conclusion and future work are presented in Section 7.

2 The JUSE-HFSS M&S Environment

The Hybrid Flow System Specification (HFSS) [3] is a formalism aimed to describe hybrid systems. HFSS combines traditional event-based systems and the novel concept of generalized sampling. HFSS can describe dense trajectories through the concept of continuous flow. These trajectories can be sampled enabling the description of numerical methods for ODE (ordinary differential equation) integration and event-detection (zero-crossing detector) [2]. HFSS has two types of models: basic and network. A basic model can read and produce continuous and discrete flows (events) and it provides the basic operators for state representation and dynamic behavior. A network model is a composition of basic models and/or other network models providing an abstraction for representing hierarchical systems. The topology (composition and linking) of HFSS networks is dynamic, enabling to adjust the model according to, for example, the distance between moving entities. It is thus possible to link mobile entities that are within sensor range, and to remove these communication links when entities become far apart. Network topology is managed by the HFSS executive component. This element is a regular HFSS basic model increased with topology management operations. This design enables the direct communication of the executive with the other network components. These components can make requests to modify the topology and also to retrieve information about the current network topology. A discussion of the advantages of this approach can be found in [6].

JUSE-HFSS is a modeling and simulation environment based on the HFSS formalism and in the JUSE component-based programming language [5]. JUSE follows closely HFSS hierarchical and modular design and it supports the development of independent and reusable software through the concept of pluggable units (PUs). Basic PUs communicate through ports avoiding the object-oriented communication that limits large-scale software reuse. PUs can be composed into networks that can be handled as basic PUs. PUs inter-communication is made through links that provide support for adapting incompatible interfaces. Figure 1 represents the `Generator` PU that produces a triangular wave whose continuous flow can be accessed through sampling, and that sends discrete flows (events) to signal maximum and minimum values. The signal can be sampled at input port `CF`, and discrete flows are sent through the output port `DF`. Generator trajectories are depicted in Figure 2. The continuous flow is *virtually continuous* since it only becomes known when the generator is sampled.

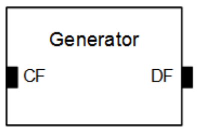

Figure 1: Generator structure.

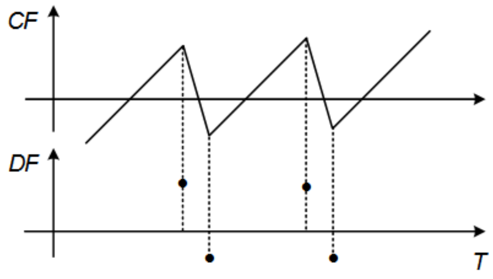

Figure 2: Generator output trajectories.

PUs can be combined hierarchically to form PU networks. Figure 3 describes a network composed by a generator and a zero-cross detector. The latter samples the environment through port `sample` and issues a signal through port `Z` when the input signal is zero. In JUSE-HFSS, zero-cross components use adaptive sampling in oder to accurately find detection times. Each HFSS network defines its own executive that is responsible to manage network topology. The executive has an open definition since it behaves like any other PU. However, since it is responsible to define the topology, it can modify the corresponding network topology reacting to incoming requests. Figure 4 represents the information received by the executive, that corresponds to the derivative of the signal when it crosses zero. In this example, we consider for simplicity a network with a static topology. Dynamic topologies are used in the next sections.

The network executive plays a similar role to the HLA Runtime Infrastructure (RTI), the main difference is that, as mentioned above, it has an open definition enabling its adaptation to the application domain. On the contrary, the RTI has a fixed interface preventing its adjustment to specific scenarios.

The topology of the Figure 3 can be described by JUSE-HFSS Listing 1. Line 4 adds a component of class `Generator` with the name `"Generator"`. Line 5 adds a component of class `ZeroCross`. The executive is declared implicitly since it is present in every network. Links between components are defined in lines [6-8]. Executive method associated with port `detect`, defined in line 10, simply prints the time and the signal value for trajectory plotting.

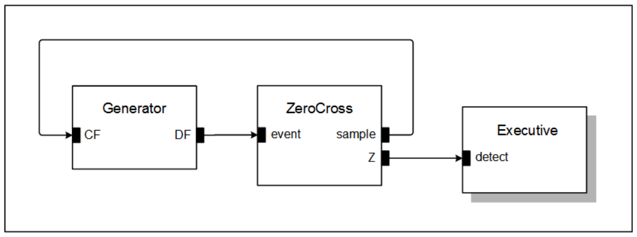

Figure 3: Generator and zero-cross detector.

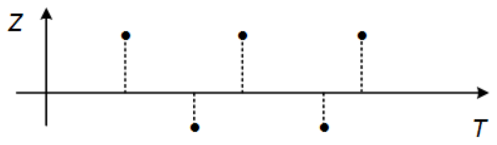

Figure 4: Zero-cross trajectory.

```
class GeneratorZeroCross extends Executive {
    public void topology() {
        super.topology();
        addComponent(Generator, "Generator");
        addComponent(ZeroCross, "ZeroCross");
        link("Generator", "DF", "ZeroCross", "event");
        link("ZeroCross", "sample", "Generator", "CF");
        link("ZeroCross", "Z", "Executive", "detect");
    }
    public void detect(double clock, double value) {println "$clock\t$value";}
}
```

Listing 1: Network topology of the `GeneratorZeroCross` model.

3 HLA Overview

The High Level Architecture (HLA) is a standard created by the US Defense Modeling and Simulation Office (DMSO) to enable the interoperability of simulators communicating through a computer network [9, 10]. HLA main advantage is to allow the interoperability of simulators to create complex scenarios that require the interaction among many entities. HLA supports two types of components, federates, that model the dynamic entities to be simulated and federations, a combination of federates. Another type of components named HLA-objects can be used to achieve the communication through shared memory. HLA-objects are passive entities depending on federates to be modified. The federates involved in HLA-object management and information retrieval have their reuse severely limited since they depend on the existence of these entities. This type of dependency jeopardizes model reuse, and in our perspective, should be avoided. We discussed in the next sections how HFSS sampling can be used to replace HLA-objects while keeping model reuse.

Federates can be considered an equivalent to HFSS basic models constrained to discrete even behavior. A federation is similar to the HFSS network where the RTI (Run-Time Infrastructure) plays the role of HFSS network executive. The HLA has several constraints that prevents to achieve the same modeling expressiveness (from a practical perspective) of the HFSS formalism.

Federates cannot be composed hierarchically, making it difficult do develop complex models. An aircraft, for example, cannot be decomposed into several components for describing its different sub-systems, and must be handled, in HLA, as a monolithic entity. The HLA provides only support for discrete event system. The support for sampling requires mostly the use of the HLA-objects mentioned above. As referred to in Section 1, we consider that sampling is fundamental for creating numerical integrators and zero-cross detectors, key components in hybrids models. HLA federate-to-federate interaction is based on publish-subscribe communication (PSC), requiring an extensive filtering support for discarding non-wanted messages. We conjecture that the choice of PSC was motivated by federate reuse since this communication protocol enables to decouple entities into independent components that can be reused in different scenarios. An aircraft component, for example, can interact with a variable number of radars without the need to modify the exiting code for achieving their interoperability. Given the architecture of the HLA, this can be considered an adequate design choice to achieve model reuse and interoperability. PSC has, however, several inherent limitations related to expressiveness and efficiency. Since PSC is a push protocol, information pulling requires two messages, one for requesting a value and other to send it. PSC is essentially a broadcast-based protocol, that requires the extensive filter computation offered by HLA data distribution management (DDM), to become more efficient. In general filters are not *exact* and many messages can be delivered to a federate that cannot actually receive them, decreasing HLA performance, since these unwanted messages need to be deliverer by the RTI, filtered by the federate and discarded. We consider that many HLA features and limitations are due to the poor design of the RTI that exhibits a fixed interface (API) and that is not a federate. In fact, this choice seems to be responsible to the lack of P2P communication, since conventional object-oriented programming would require federates to be aware of their peers jeopardizing model reuse, given that federates would no longer be independent of the topology of a specific federation. However, HFSS can guarantee model independence and reuse by locating all domain-specific information in the network executive. The executive becomes thus the only topology-dependent model while all other models can be kept independent. For example, aircrafts and radars can be kept independent and reusable while the topology is managed by the executive, a domain-specific component. Since the executive is a full-fledged component it can encode topology adaptation behavior to modify it according, for example, to the location of other components. This can be achieved by linking and unlinking components according to their proximity, offering a flexible framework to represent mobility. HFSS open executive design enables, for example, to choose the most appropriate spatial partition algorithm for a given application domain. This can be particularly relevant, since many algorithms have been developed [1, 7], while the HLA imposes a single solution. As an extreme case we have developed a representation of a mobile phone network based exclusively on P2PC without requiring spatial partitioning [4]. Another limitation of the HLA relies on its inability to provide support for the run-time creation/destruction of federates by other federates. This feature becomes very convenient to model, for example, missiles and collision detectors by independent components, as described in Section 5. In the next section we show how HFSS can support both publish-subscribe and peer-to-peer communication styles ir oder to achieve a more flexible design.

4 Combining Publish-Subscribe and Peer-to-Peer Communication

Publish/subscribe operations provide an abstraction for describing dynamic topologies. New components can be added/removed dynamically to/from a network without affecting the existing components. We show now how PSC can be described in a P2P formalism like the HFSS [3] while keeping full compatibility with P2P communication, being the overall effect an architecture able to support both communication styles simultaneously. In this description we use JUSE-HFSS a Groovy/Java implementation of the HFSS formalism described in Section 2. PSC uses publish and subscribe operators in order to achieve a simple description of broadcast topologies. While HLA PSC is achieved using the RTI as a central access point for communication exchange, HFSS uses explicit P2PC between components. The communication with the executive becomes only required to modify the peer-to-peer links. The unification of PSC and P2PC is accomplished here by considering PSC as providing a bulk specification for a peer-to-peer topology.

JUSE can define two types of publish-subscribe commands, the conventional push style associated with discrete flows/events (DF) and a new form of publish-subscribe based on continuous flows (CF). The latter enables components to pull information from several components in a single operation, providing a more effective communication, than the conventional push style, for describe sampling. Information pulling can be used, for example, to model a radar that samples the position of several drones in a single operation. Information pulling can also be used as an alternative to HLA-objects mentioned in Section 3. The topology of Figure 5 can be described in a compact manner by the bulk commands `publish(CF, xy)` and `publish(DF, comm)` issued by drone components `Drone-1`, ..., `Drone-3`, and by the commands `subscribe(CF, xy)` and `subscribe(DF, comm)` issued by component `Radar`, a radar that can sample done positions. Radar information can be used to launch a pursuer to disable the drone.

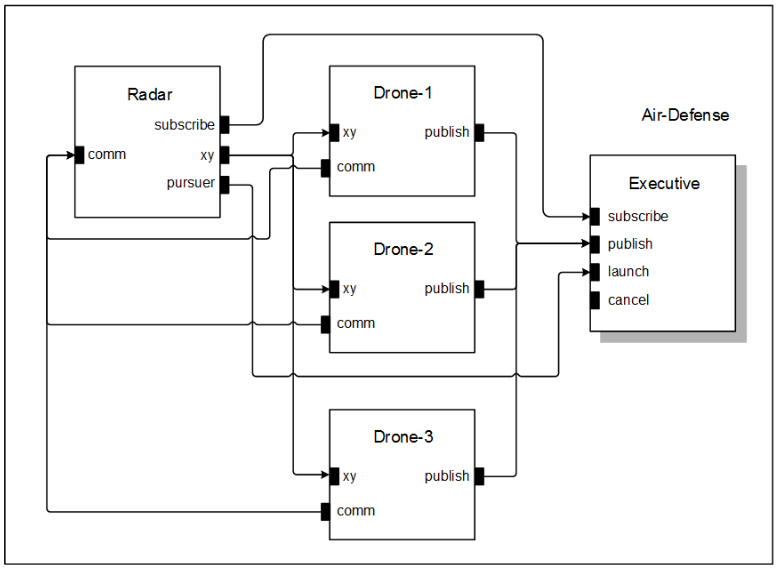

Figure 5: Representing publish-subscribe communication using peer-to-peer communication.

This compact notation simplifies topology management since links become implicitly defined. When, for example, a new drone publishes the continuous flow port `xy` it becomes automatically connected to all radars subscribing that port. Similarly, when a new radar subscribes the continuous flow port `xy` it becomes linked to all drones. The link of discrete flow ports is made through a similar procedure. When the radar subscribes to the discrete flow port `comm` it becomes connected to all the drones that have published that discrete flow. Publish-subscribe communication requires that, upon creation, drones and radars become linked to executive ports `publish` and `subscribe`.

We now present a scenario evolution after `Drone-3` has been detected by the radar and a pursuer was launched to disable this threat. Publish-subscribe operators provide no longer useful operators to represent the new topology of Figure 6. When the executive receives a `launch` signal it creates a new pursuer to follow `Drone-3`. Trying to establish a connection between these components using publish/subscribe operators would be inefficient since what is actually required is just the creation of a peer-to-peer link between radar port `xy` and `Drone-3` port `xy`. This link can be easily established by the executive that, due to its open model, can accommodate specific rules to manage particular scenarios. Another component, a zero-cross detector, is also created to find the time when the pursuer disables the done. These links are required to sample both pursuer and drone positions and to signal the executive when the drone becomes disabled. We consider that these links are too specific to be efficiently described by publish-subscribe operators, since they involve known entities without any requirement for information broadcast.

A limitation of the publish-subscribe style is that it imposes constraints on port names and on the values that can be transmitted, making it more difficult to merge components that were not designed to work together. These limitations greatly reduce the possibility of model reuse and interoperability. On the contrary, peer-to-peer communication can exploit JUSE adapter capabilities that can be used to match both port names and the

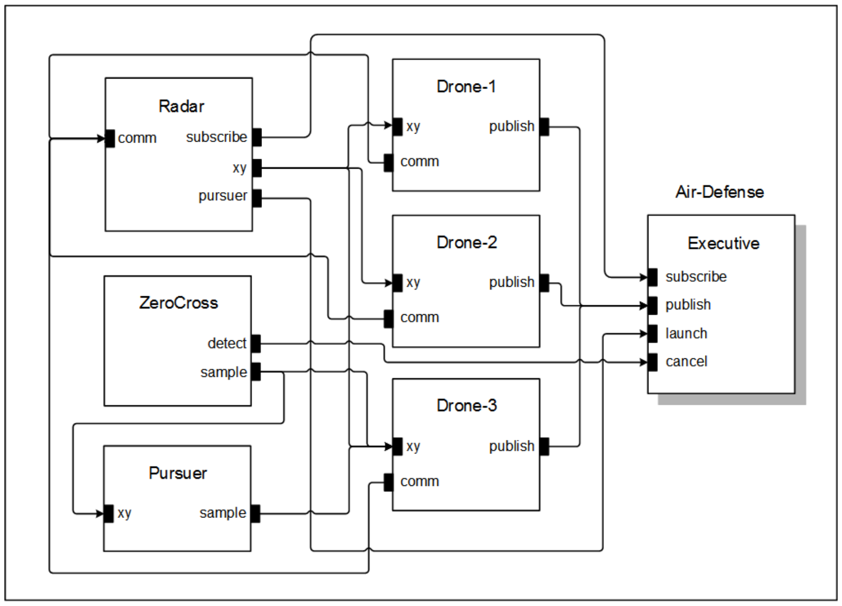

Figure 6: Combining publish-subscribe and peer-to-peer communication styles.

flow of information as shown in Section 5.

PSC, for being effective, requires spatial partitioning for modeling mobile entities, imposing an additional complexity for managing regions of interest and also a performance penalty for finding matches. These perforce penalties can be reduced by P2PC in the cases when entities have already been linked for communication. Taken for example the P2P connection used in the pair drone-pursuer, it does not require the use of regions of interest (ROIs). If, at any time, the drone becomes outside pursuer range, the pursuer can destroy itself since it has lost the target.

In the next section we present a simulation model for a more complex scenario where the simple radar is replaced by a rotating beam and where spatial partitioning is used to find the moving entities within interaction range.

5 Air-Defense Scenario

In the last section we have described the integration of publish-subscribe and peer-to-peer communication styles. We show now how spatial partitioning can be integrated with P2PC in order to achieve an efficient description of mobile entities. We consider a region of interest (ROI) manager component with the ability to keep track of publish/subscribe regions as required by space partitioning algorithms [7]. Entities can declare ROIs to the manager and, when these regions overlap, this component sends a signal to the executive that can adapt the topology in order to connect the entities that have entered in communication range. Figure 7 depicts a scenario where the three drones and the radar have declared their ROIs using ports `pubROI` and `subROI`, respectively, but there is still no overlapping between radar subscribing ROI and drones publishing ROIs. No links have been established yet between the radar and the drones.

As entities move, their regions of interest will eventually overlap. Figure 8 represents the network topology when the ROIs of the `Radar` and `Drone-4` overlap. The manager upon overlap detection sends a request to the executive that will create a link between the radar and the drone so it can be tracked. We note that `Drone-4` cannot be used in conventional publish-subscribe communication since its interface does not match radar sampling port `xy`. Additionally, drone port `xyMi` conveys the position in miles, while the radar requires

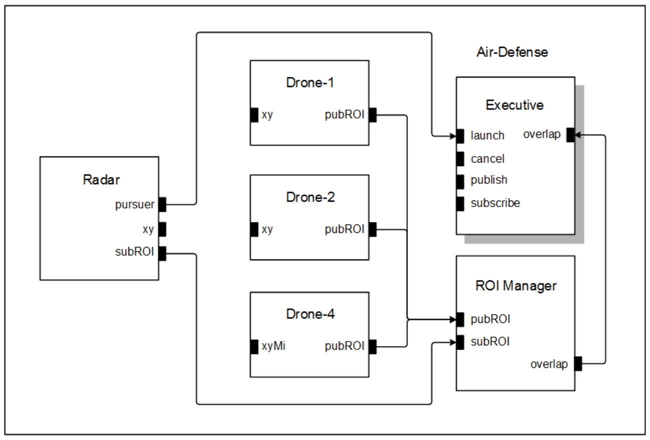

Figure 7: Using spatial partitioning for managing entity interaction.

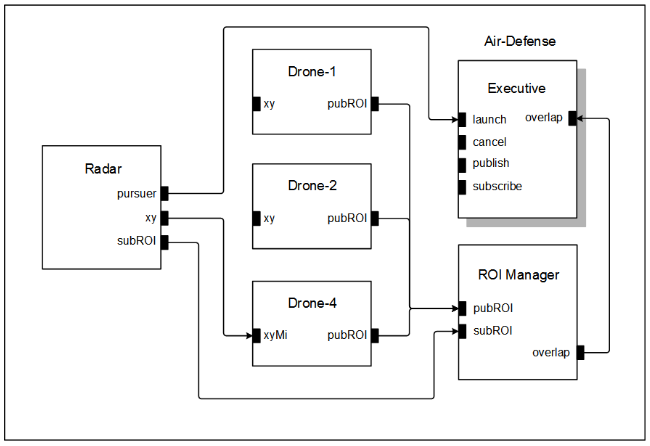

Figure 8: Drone and radar with overlapping publish/subscribe regions.

this information in km. The communication between the radar and `Drone-4` is established by the JUSE-HFSS command:

```
link("Radar","xy","Drone-4","xyMi",{x->x},{mi->[1.61*mi[0],1.61*mi[1]]})
```

that declares the identity function for the forward adapter (no need for conversion), and the conversion mi$\rightarrow$km (a 1.61 factor), for the reverse adapter, as required by the radar. We could also choose to keep PSC communication (except for `Done-4`) as described in the last section, that for simplicity has been omitted in this example. Eventuality, the radar will request the executive to launch a pursuer as described before in Figure 6. As shown in this example, HFSS dynamic peer-to-peer communication is able to represent publish-subscribe communication and also the space partitioning algorithms required to obtain an efficient representation of moving entities. Additionally, peer-to-peer links can be established using JUSE-HFSS adapter capabilities. Adapters enable to merge incompatible interfaces, enhancing model reuse. In the next section we present simulation results obtained in JUSE-HFSS for a defense scenario involving airborne rotating-beam scanning radars.

6 Simulation Results

We consider now the simulation of a more complex scenario involving two airborne radars, `R1` and `R2`, moving at constant velocities with fixed radius trajectories. There are also two drones, `D1` and `D2`, moving at a constant velocity but with a piecewise constant radius that changes at random times. The radars are modeled by their rotating beams that have a fixed period. We are interested in computing the echoes produced by the radar when detecting the two drones. In this system echoes are produced when a radar beam crosses a drone requiring, for each drone, a zero-cross detector able to compute the distance between a line (radar beam) and a point (drone). This type of interaction requires the creation of a new detector for each drone, and it benefits from P2PC to enable the detector to sample the position information from both the radar beam and from a specific drone. Upon detection, radars can launch a pursuer to disable the drone. Pursuers have a digital controller that uses a proportional law for guidance [12]. Upon collision, both pursuer are drone are removed from the simulation. The two radars coordinate so there is only one pursuer following a drone. Simulation results for this system are depicted in Figure 9, where pursuers, `P1` and `P2`, have been assigned to drones `D1` and `D2`, respectively. Results include drone, radar and pursuer trajectories. Also represented are radar echoes, and the current positions of radar, beams and drones.

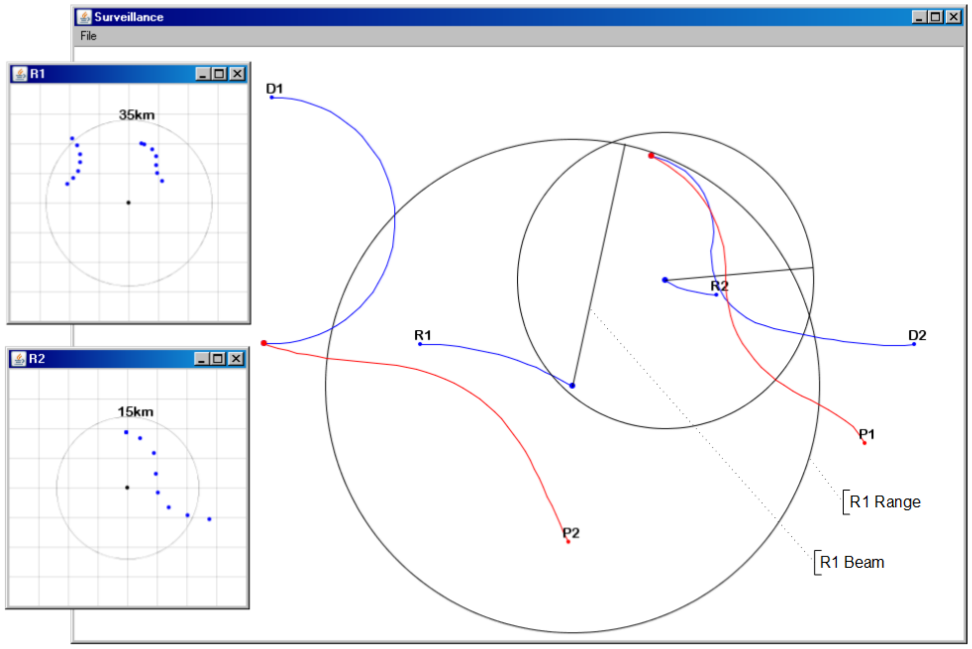

Figure 9: Simulation results for the air-defense scenario.

7 Conclusion and Future Work

We have described publish-subscribe interaction using a peer-to-peer communication network with a dynamic topology. This mapping enables both communication styles to be used in combination to describe different aspects of the same model. This integration was achieved in the JUSE-HFSS modeling and simulation environment, a framework for describing hybrid systems with a dynamic topology. While PSC can provide a short notation to specify link information, it is not well suited to describe the communication that only refers to specific entities. The integration of styles enables modelers to use the best representation for a given system.

As future work we plan to extend JUSE-HFSS model library in order to represent more complex and detailed scenarios. We consider also that future versions of the HLA supporting the modeling constructs discussed here, like the open RTI, sampling, and peer-to-peer connections, can improve its expressiveness simplifying the description of complex systems.

References

[1] L. Arge, M. de Berg, H. Haverkort, and K. Yi. The priority R-Tree: A practically efficient and worst-case optimal R-tree. *ACM Transactions on Algorithms*, 4(1), 2008.

[2] F. Barros. Dynamic structure multiparadigm modeling and simulation. *ACM Transactions on Modeling and Computer Simulation*, 13(3):259–275, 2003.

[3] F. Barros. Semantics of discrete event systems. In *Distributed Event-Based Systems*, pages 252–258, 2008.

[4] F. Barros. Modeling and simulation of mobile phones using dynamic topologies. In *Symposium on Theory of Modeling and Simulation*, 2012.

[5] F. Barros. Aspect-oriented programming and pluggable software units: a comparison based on design patterns. *Software Practice and Experience*, DOI: 10.1002/spe.2224, 2013.

[6] F. Barros. On the representation of dynamic topologies: The case for centralized and modular approaches. In *Symposium on Theory of Modeling & Simulation*, 2014.

[7] C. Ericson. *Real-Time Collision Detection*. Elsevier, 2005.

[8] E. Gamma, R. Helm, R. Johnson, and J. Vlissides. *Design Patterns: Elements of Reusable Object-Oriented Software*. Addison-Wesley, 1995.

[9] IEEE. 1516-2010 - IEEE standard for modeling and simulation high level architecture: Framework and rules. Technical report, IEEE Computer Society, 2010.

[10] F. Kuhl, R. Weatherly, and J. Dahmann. *Creating Computer Simulation Systems: An Introduction to the High Level Architecture*. Prentice Hall, 1999.

[11] K. L. Morse and J. S. Steinman. Data distribution management in the HLA: Multidimensional regions and physically correct filtering. In *Spring Simulation Interoperability Workshop*, pages 343–352, 1997.

[12] U. Shukala and P. Mahapatra. The proportional navigation dilemma - pure or true? *IEEE Transactions on Aerospace and Electronic Systems*, 26(2):382–392, 1990.